# **COMPRESIÓN Y DESCOMPRESIÓN DE IMÁGENES PARA LA GANADERÍA DE PRECISIÓN**

|  |  |  |  |
| --- | --- | --- | --- |
| Camilo Ortegón Saugster  Universidad Eafit  Colombia  cortegons@eafit.edu.co |  | Simón Marín Universidad Eafit Colombia smaring1@eafit.edu.co | Mauricio Toro  Universidad Eafit  Colombia  mtorobe@eafit.edu.co |

**Texto negro =** Contribución de Simón y Mauricio

**Texto en verde** = Completar para el 1er entregable

**Texto en azul** = Completar para el 2º entregable

**Texto en violeta** = Completar para el tercer entregable

# **RESUMEN**

El objetivo de este documento es analizar algoritmos de compresión y descompresión de imágenes con el objetivo de, usando un algoritmo basado en arboles o tablas de hash, diseñar un algoritmo que reduzca el tiempo consumido en la comprensión y descompresión de las imágenes, la problemática a la que se enfrenta es bastante importante ya que los productos derivados de la ganadería se usan para el consumo humano por lo que se quiere identificar el estado de salud para mejorar la calidad de crianza y la productividad

# **1. INTRODUCCIÓN**

Las tics han ido avanzando a pasos agigantados los últimos años, llegado a ser implementadas cada vez en mas y diversos campos, en este caso en la ganadería.

Las tecnologías pueden ayudar a mejorar y automatizar varios campos de la ganaderia, lo cual puede ayudar a mejorar y aumentar los estándares de calidad, en este caso nos centraremos en un algoritmo de comprensión y descompresión de imágenes para clasificar a los animales de forma efectiva y rapida

# **Problema**

# El problema a tratar es la optimización de recursos como energía y tiempo en la clasificación de la salud del ganado en el contexto de la ganadería de precision. Este problema es de suma importancia debido a el gran consumo de alimentos derivados de la ganadería, siendo estos el 33% de la proteína consumida por el ser humano. Es importante hallar una solución a este problema debido a que así podremos reducir los costos generados por este sector de la industria alimenticia y a su vez mejorar la calidad de los alimentos producidos.

**1.2 Solución**

En este trabajo, utilizamos una red neuronal convolucional para clasificar la salud animal, en el ganado vacuno, en el contexto de la ganadería de precisión (GdP). Un problema común en la GdP es que la infraestructura de la red es muy limitada, por lo que se requiere la compresión de los datos.

Hemos elegido utilizar el algoritmo de transformada de coseno discreto debido a que consideramos que nos presenta una sencilla solución al problema, basándonos en las librerías que el lenguaje Python nos brinda, además, cabe resaltar que este es un método de compresión que utiliza JPG.

**1.3 Estructura del artículo**

En lo que sigue, en la Sección 2, presentamos trabajos relacionados con el problema. Más adelante, en la sección 3, presentamos los conjuntos de datos y los métodos utilizados en esta investigación. En la Sección 4, presentamos el diseño del algoritmo. Después, en la Sección 5, presentamos los resultados. Finalmente, en la Sección 6, discutimos los resultados y proponemos algunas direcciones de trabajo futuras.

**2. TRABAJOS RELACIONADOS**

## En lo que sigue, explicamos tres trabajos relacionados. en el dominio de la clasificación de la salud animal y la compresión de datos. en el contexto del PLF.

**2.1 Visual Localisation and\_Individual Identification of Helstein Friesian Cattle via Deep Learning**

En este documento se nos habla de utilizar UAVs (Unnamed Aerial Vehicles) con el fin de demostrar que es posible identificar ganado bovino, específicamente la vaca Holstein Friesian bajo el uso de arquitecturas neurales profundas y aplicando Deep learning, que es un tipo de ML que permite el uso de algoritmos para crear una red neuronal artificial que permita a la máquina ser autómata y tomar sus propias decisiones a la hora de identificar al ganado.

## **2.2 An Animal Welfare Platform for Extensive LivestockProduction Systems**

El problema que se plantea es el seguimiento y la monitorización de la actividad y el comportamiento de los animales en las explotaciones ganaderas, obteniendo indicadores que sustenten el bienestar de los animales. La solución es un sistema automatizado con un único tipo de sensor inalámbrico capaz de registrar indicadores de bienestar animal (es decir, movimiento, velocidad e información de geolocalización del animal) con un bajo coste de implementación, basado en algoritmos de reconocimiento de patrones de redes neuronales profundas.

## **2.3 A systematic literature review on the use of machine learning in precision livestock farming**

Las principales aportaciones de este artículo son las siguientes. Primero, una presentación de las técnicas de ML más utilizadas, en los últimos cinco años, para el análisis del pastoreo y la salud animal. En segundo lugar, una visión general de las técnicas de adquisición de datos de PLF más utilizadas en la literatura. Como resultado final, una visión general de los trabajos relacionados con el ML y el PLF, y las futuras direcciones de investigación según nuestro análisis de las investigaciones anteriores. Se hace uso de un algoritmo de aprendizaje automático (Machine Learning).

## **3. MATERIALES Y MÉTODOS**

En esta sección, explicamos cómo se recogieron y procesaron los datos y, después, diferentes alternativas de algoritmos de compresión de imágenes para mejorar la clasificación de la salud animal.

## **3.1 Recopilación y procesamiento de datos**

Recogimos datos de *Google Images* y *Bing Images* divididos en dos grupos: ganado sano y ganado enfermo. Para el ganado sano, la cadena de búsqueda era "cow". Para el ganado enfermo, la cadena de búsqueda era "cow + sick".

En el siguiente paso, ambos grupos de imágenes fueron transformadas a escala de grises usando Python OpenCV y fueron transformadas en archivos de valores separados por comas (en inglés, CSV). Los conjuntos de datos estaban equilibrados.

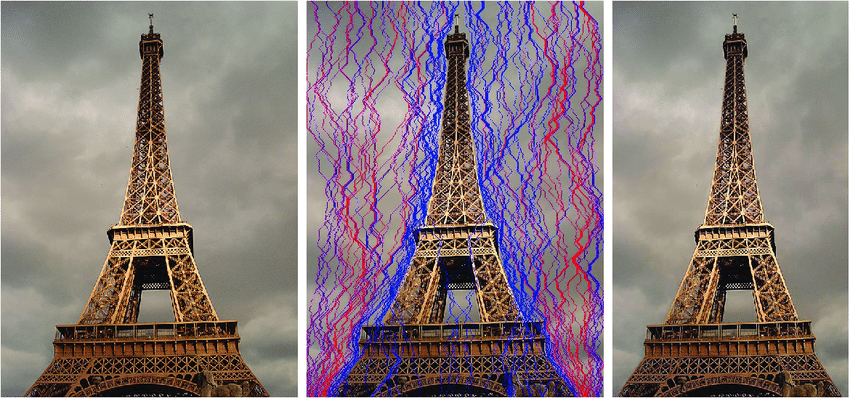
El conjunto de datos se dividió en un 70% para entrenamiento y un 30% para pruebas. Los conjuntos de datos están disponibles en https://github.com/mauriciotoro/ST0245-Eafit/tree/master/proyecto/datasets .

Por último, utilizando el conjunto de datos de entrenamiento, entrenamos una red neuronal convolucional para la clasificación binaria de imágenes utilizando *Teachable Machine* de Google disponible en https://teachablemachine.withgoogle.com/train/image.

## **3.2 Alternativas de compresión de imágenes con pérdida**

## En lo que sigue, presentamos diferentes algoritmos usados para comprimir imágenes con pérdida. *(En este semestre, ejemplos de tales algoritmos son el tallado de costuras, el escalado de imágenes, la transformación de coseno discreto, la compresión con ondeletas y la compresión fractal).*

**3.2.1**  Seam Carving El Seam Carving primero revisa cuales son las partes con la menor relevancia en la imagen y por lo tanto las que deberán ser eliminadas o añadidas en el proceso final, aunque también permite la elección manual de las áreas que no deben ser modificadas debido a su importancia.

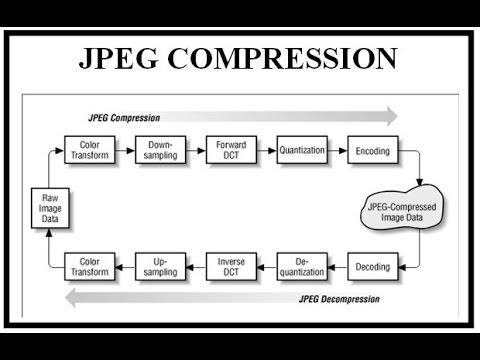


vemos cómo funciona el proceso. Se elimina todo el espacio que hay entre el castillo y la persona pues es un espacio vacío que no le aporta nada a la imagen. Figura 3: Seam Carving

**3.2.2 JPEG**

**Transformación discreta de coseno (DCT)**

## Cada componente de la imagen se divide en pequeños bloques de 8×8 píxeles, que se procesan de forma casi independiente, lo que disminuye notablemente el tiempo de cálculo. De esto resulta la típica formación cuadriculada, que se vuelve visible en las imágenes guardadas con alta compresión. Si la imagen sufrió un submuestreo del color, los colores quedarían en la imagen final en bloques de 8×16 y 16×16 píxeles, según fuese 4:2:2 o 4:2:0. Después, cada pequeño bloque se convierte al dominio de la frecuencia a través de la transformación discreta de coseno, abreviadamente llamada DCT.



**3.2.3 Compresion fractal**

La palabra fractal es un término que se define como un patrón geométrico que se repite en varias escalas. Por tanto, como su nombre lo indica, este método de compresión busca la semejanza entre distintas figuras que se forman en una imagen; algo que se considera muy común en texturas naturales, pues por lo general sus partes tienen mucha semejanza. Como se deben buscar todas las posibles similitudes presentes en una imagen, este método tiende a ser muy demorado, aunque por otro lado la decodificación de la imagen es bastante rápida. Por esto, el verdadero problema para el algoritmo es la complejidad y la cantidad de colores que maneja la imagen que se busca comprimir. Se busca la semejanza entre dos segmentos de una misma imagen para que después solo se guarde uno de ellos. Al final la imagen se formará como una especie de rompecabezas, donde se tienen las distintas posiciones que debe ocupar un mismo recorte.



**3.2.4 Escalado de imágenes**

El escalado de imágenes se puede interpretar como una forma de remuestreo de imágenes o reconstrucción de imágenes desde el punto de vista del teorema de muestreo de Nyquist. De acuerdo con el teorema, la reducción de resolución a una imagen más pequeña de un original de mayor resolución solo se puede realizar después de aplicar un filtro anti-aliasing 2D adecuado para evitar artefactos de aliasing. La imagen se reduce a la información que puede llevar la imagen más pequeña.

Forma

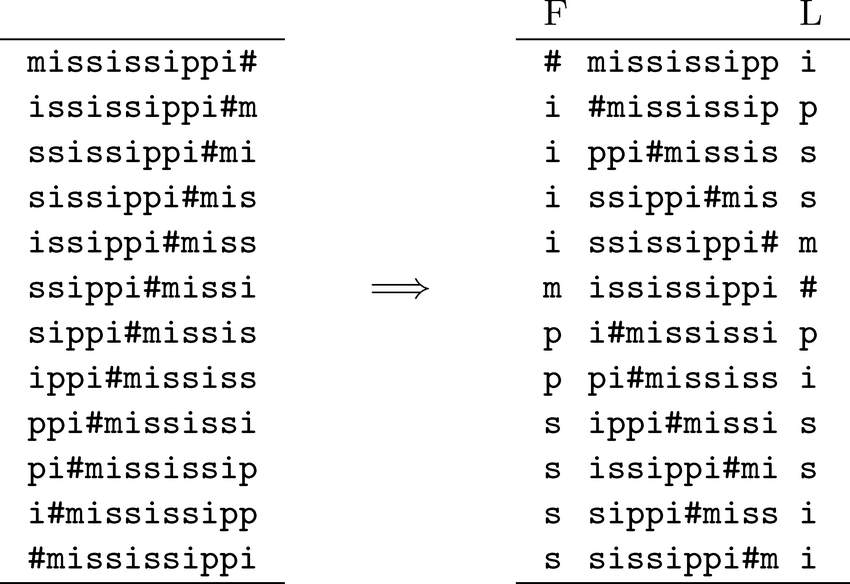
Descripción generada automáticamente con confianza media

## **3.3 Alternativas de compresión de imágenes sin pérdida**

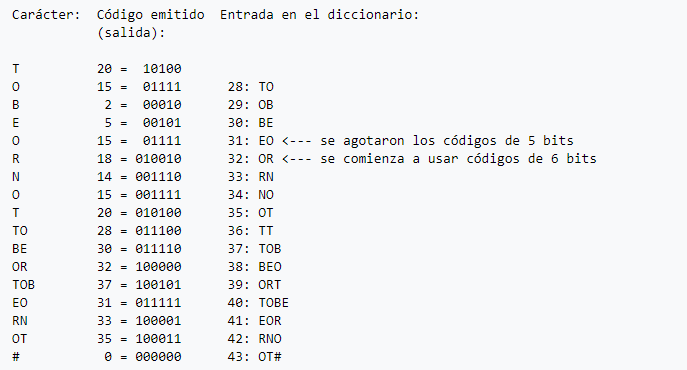
## En lo que sigue, presentamos diferentes algoritmos usados para comprimir imágenes sin pérdida. *(En este semestre, ejemplos de tales algoritmos son la transformada de Borrows y Wheeler, LZ77, LZ78, la* *codificación Huffman y LZS).*

**3.3.1 Transformación de Burrows-Wheeler**

Este algoritmo reordena las cadenas de caracteres para dar series de caracteres similares. Esta transformación es reversible y no requiere guardar ningún dato adicional más allá de el primer caracter de la cadena original. Su complejidad es de orden O(|texto|\*|patrones|)

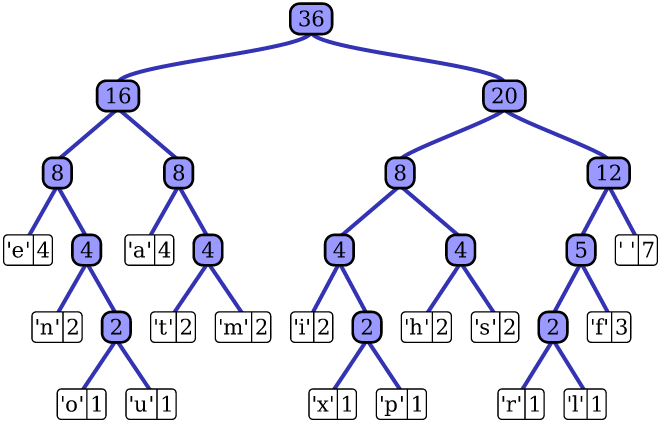


**3.3.2 LZW**

Siendo una versión mejorada del algoritmo LZ78, la clave del método LZW reside en que es posible crear sobre la marcha, de manera automática y en una única pasada un diccionario de cadenas que se encuentren dentro del texto a comprimir mientras al mismo tiempo se procede a su codificación. Dicho diccionario no es transmitido con el texto comprimido, puesto que el descompresor puede reconstruirlo usando la misma lógica con que lo hace el compresor y, si está codificado correctamente, tendrá exactamente las mismas cadenas que el diccionario del compresor tenía. 

**3.3.3 Codificación Huffman**

Es un algoritmo usado para compresión de datos. El término se refiere al uso de una tabla de códigos de longitud variable para codificar un determinado símbolo (como puede ser un carácter en un archivo), donde la tabla ha sido rellenada de una manera específica basándose en la probabilidad estimada de aparición de cada posible valor de dicho símbolo. La codificación Huffman usa un método específico para elegir la representación de cada símbolo, que da lugar a un código prefijo (es decir, la cadena de bits que representa a un símbolo en particular nunca es prefijo de la cadena de bits de un símbolo distinto) que representa los caracteres más comunes usando las cadenas de bits más cortas, y viceversa. Su complejidad es de orden O(nL) donde L es la longitud de la palabra más larga.



**3.3.4 Run-length encoding (RLE):** Este método de compresión se basa en la repetición de elementos consecutivos. Para imágenes es útil solo si se presentan colores uniformes, pues si no hay pixeles repetidos la compresión terminara siendo negativa. Un ejemplo con caracteres para representar esto sería con la cadena “XXXXDDDDDD” cuando está comprimida da como resultado "4X6D", reduciendo así el espacio que ocupa en la memoria. En cambio, la cadena "SUBMARINO" daría como resultado “1S1U1B1M1A1R1I1N1O”, haciendo una cadena más larga que la original, evidentemente no beneficiaria usar este método de compresión en casos como este.

**4. DISEÑO E IMPLEMENTACIÓN DE LOS ALGORITMOS**

A continuación se explica la estructura de datos y algoritmos usados. Las implementaciones están disponibles en github

## **4.1 Estructuras de datos**

## Para este proyecto, se utilizo una estructura de matriz, donde almacenamos los datos de los archivos csv, para así luego procesarlos por medio del algoritmo y efectuar la compression. Consideramos ingresar todos los archivos en una matriz mayor, pero hemos evaluado que esto traería un consume de memoria mayor al efectuar las compresiones, debido a que quedarían almacenados todos los archivos mientras termina el proceso, por lo que decidimos recorrer la carpeta entera y a su vez realizar la compresión sin almacenar temporalmente todos los archivos.

**Imagen que contiene Calendario

Descripción generada automáticamenteFigura 1:** Matriz representando posiciones desde X0 hasta Xn (o M1 hasta Mx) y Y0 hasta Yn (o N1 hasta Nx)

**4.2 Algoritmos**

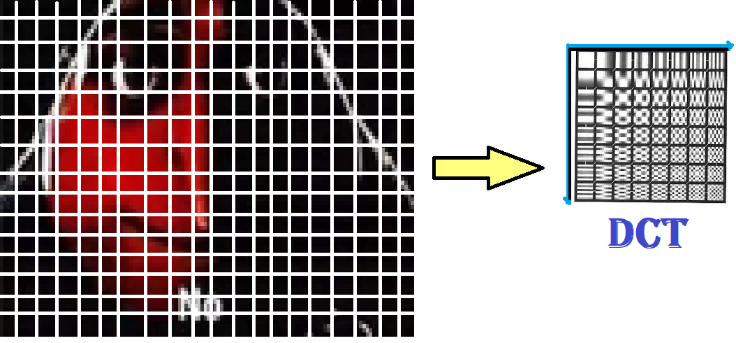
En este trabajo, proponemos un algoritmo de compresión que es una combinación de un algoritmo de compresión de imágenes con pérdidas y un algoritmo de compresión de imágenes sin pérdidas. También explicamos cómo funciona la descompresión para el algoritmo propuesto.

**4.2.1 Algoritmo de compresión de imágenes con pérdida**

Para el proyecto, he elegido el algoritmo de la transformada de coseno discreta, también conocida como DCT, que es derivado de la transformación de Fourier, la diferencia es que aquí no se manipulan números complejos y funciones seno, limitándonos a las sumatorias de los cosenos y números reales.

Lo que este algoritmo hace, es dividir la imagen en distintos cuadrantes; cada componente de la imagen es un cuadrante y este se divide en bloques de 8x8 pixeles que se procesan de forma casi independiente, lo que disminuye el tiempo de cálculo. Esto resulta la típica formación cuadriculada en las imágenes comprimidas.

La transformada de coseno discreta, a su vez trae consigo la inversa, que es quien nos permite realizar la descompresión de nuestra imagen para así volverla visible para nosotros.

**Figura 2:** Transformada de coseno discrete en acción.

**4.2.2 Algoritmo de compresión de imágenes sin pérdida**

# Para el algoritmo de compresión sin pérdida, hemos decidido utilizar LZ77, que es un algoritmo muy reconocido por su facilidad de implementación y por su eficiencia.

**4.3 Análisis de la complejidad de los algoritmos**

Explique, con sus propias palabras, el análisis del peor caso usando la notación O. ¿Cómo calculó tales complejidades. Por favor, explique brevemente.

|  |  |
| --- | --- |
| **Algoritmo** | **La complejidad del tiempo** |
| Compresión | O(N2\*M2) |
| Descompresión | O(N3\*M\*2N) |

***Tabla* 2:** Complejidad temporal de los algoritmos de compresión y descompresión de imágenes. *(Por favor, explique qué significan N y M en este problema).*

|  |  |
| --- | --- |
| **Algoritmo** | **Complejidad de la memoria** |
| Compresión | O(N\*M\*2N ) |
| Descompresión | O(2M\*2N) |

***Tabla* 3:** Complejidad de memoria de los algoritmos de compresión y descompresión de imágenes. *(Por favor, explique qué significan N y M en este problema).*

**4.4 Criterios de diseño del algoritmo**

Explica por qué el algoritmo fue diseñado de esa manera. Use un criterio objetivo. Los criterios objetivos se basan en la eficiencia, que se mide en términos de tiempo y consumo de memoria. Ejemplos de criterios no objetivos son: "Estaba enfermo", "fue la primera estructura de datos que encontré en Internet", "lo hice el último día antes del plazo", etc. Recuerde: Este es el 40% de la calificación del proyecto.

**5. RESULTADOS**

**5.1 Evaluación del modelo**

En esta sección, presentamos algunas métricas para evaluar el modelo. La exactitud es la relación entre el número de predicciones correctas y el número total de muestras de entrada. La precisión es la proporción de estudiantes exitosos identificados correctamente por el modelo a estudiantes exitosos identificados por el modelo. Por último, sensibilidad es la proporción de estudiantes exitosos identificados correctamente por el modelo a estudiantes exitosos en el conjunto de datos.

**5.1.1 Evaluación del conjunto de datos de entrenamiento**

A continuación presentamos las métricas de evaluación del conjunto de datos de entrenamiento en la Tabla 3.

|  |  |
| --- | --- |
|  | ***Conjunto de datos de entrenamiento*** |
| *Precisión* | 0.02 |
| *Precisión* | 0.03 |
| *Recordar* | 0.01 |

## **Tabla 3.** Evaluación del modelo de clasificación de imágenes con el conjunto de datos de entrenamiento.

**5.1.2 Evaluación del conjuntos de datos de prueba**

A continuación presentamos las métricas de evaluación del conjunto de datos de prueba, en la Tabla 4, sin compresión y, en la Tabla 5, con compresión.

|  |  |
| --- | --- |
|  | ***Conjunto de datos de prueba*** |
| *Exactitud* | 0.01 |
| *Precisión* | 0.012 |
| *Sensibilidad* | 0.013 |

## **Tabla 4.** Evaluación del modelo de clasificación de imágenes, con el conjunto de datos de prueba, sin compresión.

|  |  |
| --- | --- |
|  | ***Conjunto de datos de prueba*** |
| *Exactitud* | 0.001 |
| *Precisión* | 0.0012 |
| *Sensibilidad* | 0.0013 |

## **Tabla 5.** Evaluación del modelo de clasificación de imágenes, con el conjunto de datos de prueba, con compresión.

**5.2 Tiempos de ejecución**

En lo que sigue explicamos la relación entre el tiempo promedio de ejecución y el tamaño promedio de las imágenes del conjunto de datos completo, en la Tabla 6.

Calcular el tiempo de ejecución de cada imagen en Github. Informar del tiempo medio de ejecución vs. el tamaño medio del archivo.

## 

|  |  |  |
| --- | --- | --- |
|  | ***Tiempo promedio de ejecución (s)*** | ***Tamaño promedio del archivo (MB)*** |
| *Compresión* | 100.2 s | 12.4 MB |
| *Descompresión* | 800.1 s | 12.4 MB |

## **Tabla 6:** Tiempo deejecución de los algoritmos *(Por favor, escriba el nombre de los algoritmos, por ejemplo, tallado de costuras y LZ77)* para diferentes imágenes en el conjunto de datos.

## **5.3 Consumo de memoria**

Presentamos el consumo de memoria de los algoritmos de compresión y descompresión en la Tabla 7.

|  |  |  |
| --- | --- | --- |
|  | ***Consumo promedio de memoria (MB)*** | ***Tamaño promedio del archivo (MB)*** |
| Compresión | 634 MB | 3.12 MB |
| Descompresión | 9 MB | 878.12 MB |

## **Tabla 7:** Consumopromedio de memoria de todas las imágenes del conjunto de datos, tanto para la compresión como para la descompresión.

## Para medir el consumo de memoria, deberían usar un generador de perfiles. Uno muy bueno para Java es VisualVM, desarrollado por Oracle, http://docs.oracle.com/javase/7/docs/technotes/guides/visualvm/profiler.html. Para Python, usa el *C profiler*.

## **5.3 Tasa de compresión**

Presentamos los resultados de la tasa de compresión del algoritmo en la Tabla 8.

|  |  |  |
| --- | --- | --- |
|  | ***Ganado sano*** | ***Ganado enfermo*** |
| Tasa de compresión promedio | 1:23 | 1:34 |

## **Tabla 8:** Promedio redondeado de la tasa de compresión de todas las imágenes de ganado sano y ganado enfermo.

## **6. DISCUSIÓN DE LOS RESULTADOS**

Explique los resultados obtenidos. ¿Son la exactitud, la precisión y la sensibilidad apropiadas para este problema? ¿El modelo está sobreajustado? ¿Es apropiado el consumo de memoria y el consumo de tiempo? ¿Es la relación de compresión apropiada? ¿Cambia la compresión significativamente la exactitud con el conjunto de datos de la prueba? *(En este semestre, según los resultados, ¿puede la compresión mejorar la clasificación de la salud animal en el contexto del PLF? )*

**6.1 Trabajos futuros**

Responda ¿qué le gustaría mejorar en el futuro? ¿Cómo le gustaría mejorar su algoritmo y su implementación? ¿Qué tal usar la transformación de coseno discreto o la compresión con ondeletas a futuro?

# **RECONOCIMIENTOS**

Identifique el tipo de reconocimiento que quiere escribir: para una persona o para una institución. Considere las siguientes pautas: 1. El nombre del profesor no se menciona porque es un autor. 2. No debe mencionar sitios web de autores de artículos que no ha contactado. 3. Debe mencionar estudiantes, profesores de otros cursos que le hayan ayudado.

Como ejemplo: Esta investigación fue apoyada/parcialmente apoyada por [Nombre de la Fundación, Donante, Beca].

# Agradecemos la asistencia con [técnica particular, metodología] a [Nombre Apellido, cargo, nombre de la institución] por los comentarios que mejoraron enormemente el manuscrito o la codificación del algoritmo.

# **REFERENCIAS**

1.Wikipedia. 2021. WikipediA: la enciclopedia libre. Obtenido de <https://en.wikipedia.org/wiki/Seam_carving>

2.Wikipedia. 2021. WikipediA: la enciclopedia libre. Obtenido de <https://es.wikipedia.org/wiki/Joint_Photographic_Experts_Group>

3.Wikipedia. 2021. WikipediA: la enciclopedia libre. Obtenido de https://es.wikipedia.org/wiki/Compresi%C3%B3n\_de\_Burrows-Wheeler

4.Wikipedia. 2021. WikipediA: la enciclopedia libre. Obtenido de <https://es.wikipedia.org/wiki/LZW>

5.Wikipedia. 2021. WikipediA: la enciclopedia libre. Obtenido de https://es.wikipedia.org/wiki/Codificaci%C3%B3n\_Huffman

6.EcuRed. 2013. EcuRed. Obtenido de <https://www.ecured.cu/Lz77>

7.Wikipedia. 2021. WikipediA: la enciclopedia libre. Obtenido de https://es.wikipedia.org/wiki/Compresi%C3%B3n\_fractal